**Experiment 9**

Valiveti Manikanta Bhuvanesh

19BCD7088

AIM: Write a program to perform Movie recommendation system

**Code:**

library(ggplot2)

library(data.table)

library(reshape2)

library(recommenderlab)

movie\_data <- read.csv("movies.csv",stringsAsFactors=FALSE)

rating\_data <- read.csv("ratings.csv")

head(movie\_data)

head(rating\_data)

str(movie\_data)

summary(movie\_data)

str(rating\_data)

summary(rating\_data)

movie\_genre <- as.data.frame(movie\_data$genres, stringsAsFactors=FALSE)

head(movie\_genre)

library(data.table)

movie\_genre2 <- as.data.frame(tstrsplit(movie\_genre[,1], '[|]',

type.convert=TRUE),

stringsAsFactors=FALSE)

colnames(movie\_genre2) <- c(1:10)

list\_genre <- c("Action", "Adventure", "Animation", "Children",

"Comedy", "Crime","Documentary", "Drama", "Fantasy",

"Film-Noir", "Horror", "Musical", "Mystery","Romance",

"Sci-Fi", "Thriller", "War", "Western")

genre\_mat1 <- matrix(0,10330,18)

genre\_mat1[1,] <- list\_genre

colnames(genre\_mat1) <- list\_genre

for (index in 1:nrow(movie\_genre2)) {

for (col in 1:ncol(movie\_genre2)) {

gen\_col = which(genre\_mat1[1,] == movie\_genre2[index,col])

genre\_mat1[index+1,gen\_col] <- 1

}

}

genre\_mat2 <- as.data.frame(genre\_mat1[-1,], stringsAsFactors=FALSE)

for (col in 1:ncol(genre\_mat2)) {

genre\_mat2[,col] <- as.integer(genre\_mat2[,col])

}

str(genre\_mat2)

head(genre\_mat2)

SearchMatrix <- cbind(movie\_data[,1:2], genre\_mat2[])

head(SearchMatrix)

ratingMatrix <- dcast(rating\_data, userId~movieId, value.var = "rating", na.rm=FALSE)

ratingMatrix <- as.matrix(ratingMatrix[,-1])

ratingMatrix <- as(ratingMatrix, "realRatingMatrix")

head(ratingMatrix)

head(rating\_data)

recommendation\_model <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")

names(recommendation\_model)

lapply(recommendation\_model, "[[", "description")

recommendation\_model$IBCF\_realRatingMatrix$parameters

similarity\_mat <- similarity(ratingMatrix[1:4, ],method = "cosine",which = "users")

as.matrix(similarity\_mat)

image(as.matrix(similarity\_mat), main = "User's Similarities")

movie\_similarity <- similarity(ratingMatrix[, 1:4], method ="cosine", which = "items")

as.matrix(movie\_similarity)

image(as.matrix(movie\_similarity), main = "Movies similarity")

rating\_values <- as.vector(ratingMatrix@data)

unique(rating\_values)

Table\_of\_Ratings <- table(rating\_values)

Table\_of\_Ratings

library(ggplot2)

movie\_views <- colCounts(ratingMatrix)

table\_views <- data.frame(movie = names(movie\_views),views = movie\_views)

table\_views <- table\_views[order(table\_views$views,decreasing = TRUE), ]

table\_views$title <- NA

for (index in 1:10325){

table\_views[index,3] <- as.character(subset(movie\_data,movie\_data$movieId == table\_views[index,1])$title)

}

table\_views[1:6,]

ggplot(table\_views[1:6, ], aes(x = title, y = views)) +

geom\_bar(stat="identity", fill = 'steelblue') +

geom\_text(aes(label=views), vjust=-0.3, size=3.5) +

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +

ggtitle("Total Views of the Top Films")

image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")

movie\_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,colCounts(ratingMatrix) > 50]

movie\_ratings

minimum\_movies<- quantile(rowCounts(movie\_ratings), 0.98)

minimum\_users <- quantile(colCounts(movie\_ratings), 0.98)

image(movie\_ratings[rowCounts(movie\_ratings) > minimum\_movies,colCounts(movie\_ratings) > minimum\_users],main = "Heatmap of the top users and movies")

average\_ratings <- rowMeans(movie\_ratings)

qplot(average\_ratings, fill=I("steelblue"), col=I("red")) +ggtitle("Distribution of the average rating per user")

normalized\_ratings <- normalize(movie\_ratings)

sum(rowMeans(normalized\_ratings) > 0.00001)

image(normalized\_ratings[rowCounts(normalized\_ratings) > minimum\_movies,colCounts(normalized\_ratings) > minimum\_users],main = "Normalized Ratings of the Top Users")

binary\_minimum\_movies <- quantile(rowCounts(movie\_ratings), 0.95)

binary\_minimum\_users <- quantile(colCounts(movie\_ratings), 0.95)

good\_rated\_films <- binarize(movie\_ratings, minRating = 3)

image(good\_rated\_films[rowCounts(movie\_ratings) > binary\_minimum\_movies,colCounts(movie\_ratings) > binary\_minimum\_users],main = "Heatmap of the top users and movies")

sampled\_data<- sample(x = c(TRUE, FALSE),size = nrow(movie\_ratings),replace = TRUE,prob = c(0.8, 0.2))

training\_data <- movie\_ratings[sampled\_data, ]

testing\_data <- movie\_ratings[!sampled\_data, ]

library(recommenderlab)

recommendation\_system <- recommenderRegistry$get\_entries(dataType ="realRatingMatrix")

recommendation\_system$IBCF\_realRatingMatrix$parameters

recommen\_model <- Recommender(data = training\_data,method = "IBCF",parameter = list(k = 30))

recommen\_model

class(recommen\_model)

model\_info <- getModel(recommen\_model)

class(model\_info$sim)

dim(model\_info$sim)

top\_items <- 20

image(model\_info$sim[1:top\_items, 1:top\_items],main = "Heatmap of the first rows and columns")

sum\_rows <- rowSums(model\_info$sim > 0)

table(sum\_rows)

sum\_cols <- colSums(model\_info$sim > 0)

qplot(sum\_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")

top\_recommendations <- 10

predicted\_recommendations <- predict(object = recommen\_model,

newdata = testing\_data,

n = top\_recommendations)

predicted\_recommendations

user1 <- predicted\_recommendations@items[[1]]

movies\_user1 <- predicted\_recommendations@itemLabels[user1]

movies\_user2 <- movies\_user1

for (index in 1:10){

movies\_user2[index] <- as.character(subset(movie\_data,

movie\_data$movieId == movies\_user1[index])$title)

}

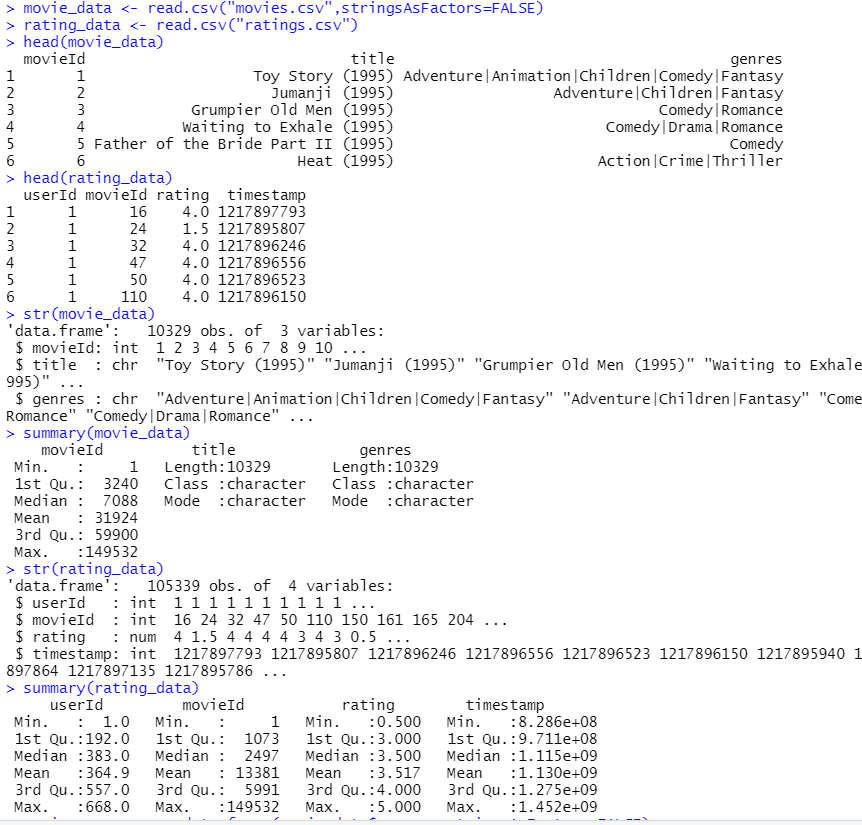
movies\_user2

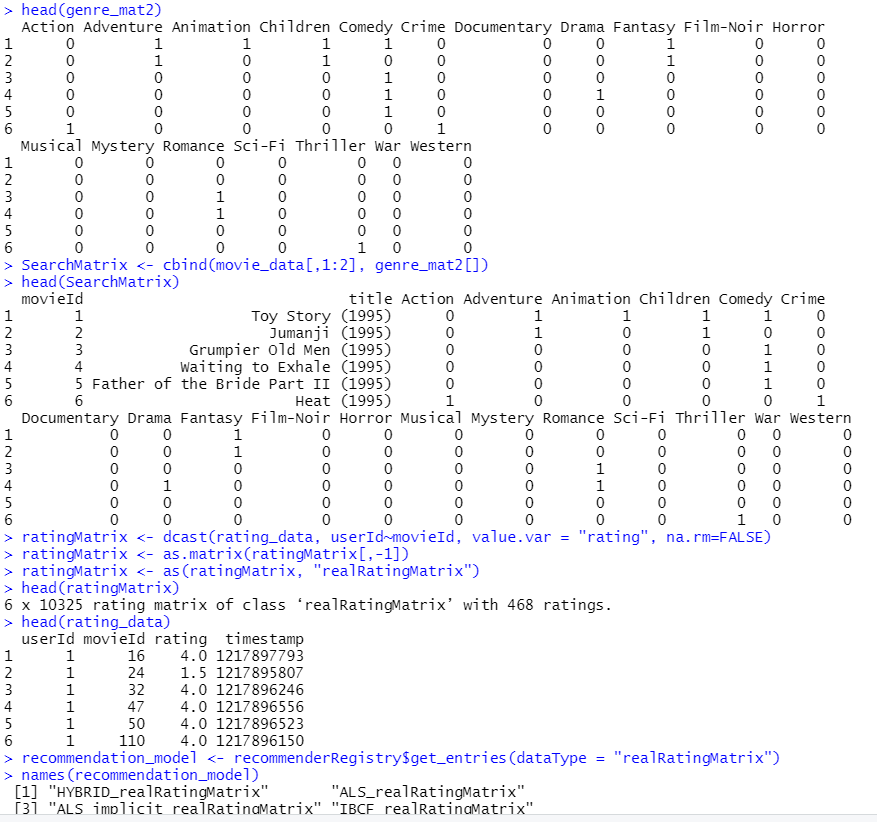
recommendation\_matrix <- sapply(predicted\_recommendations@items,

function(x){ as.integer(colnames(movie\_ratings)[x]) }) # matrix with the recommendations for each user

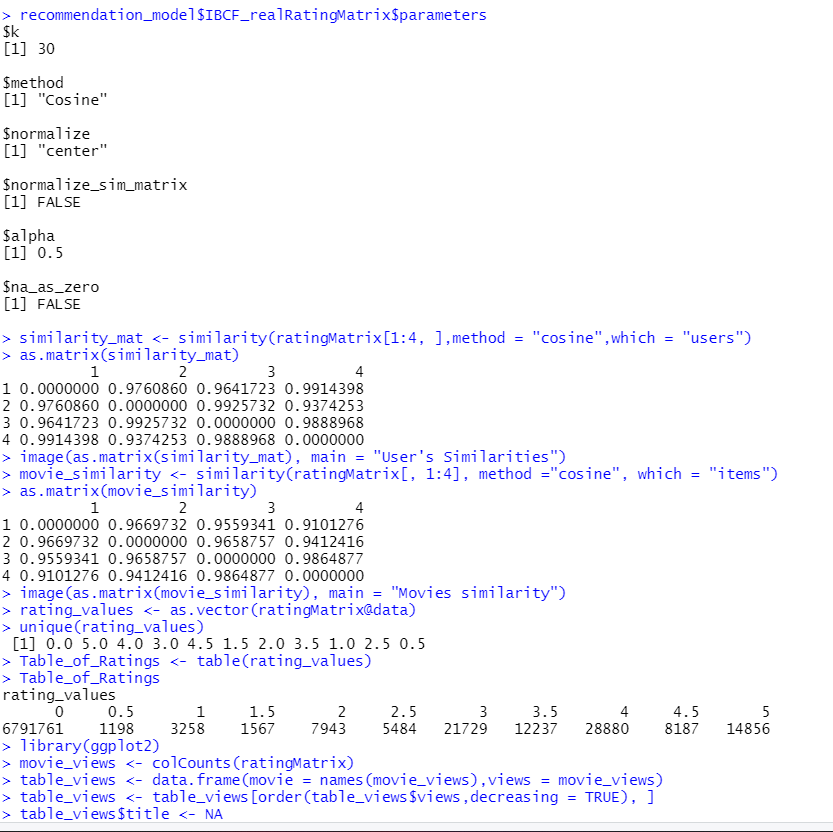
recommendation\_matrix[,1:5]

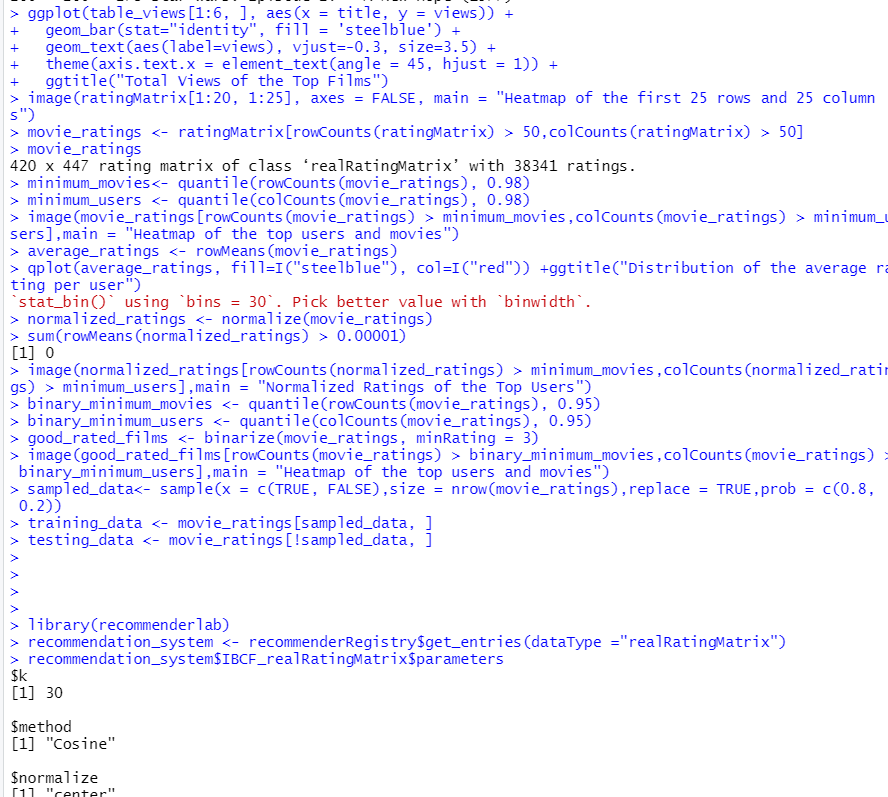
**OUTPUT:**

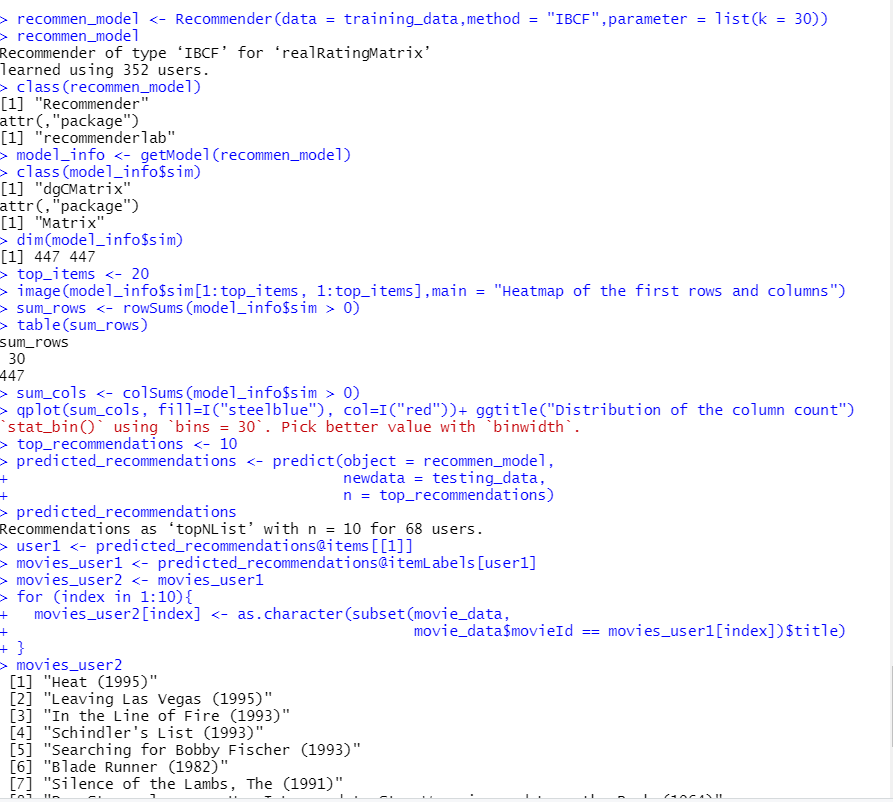
****

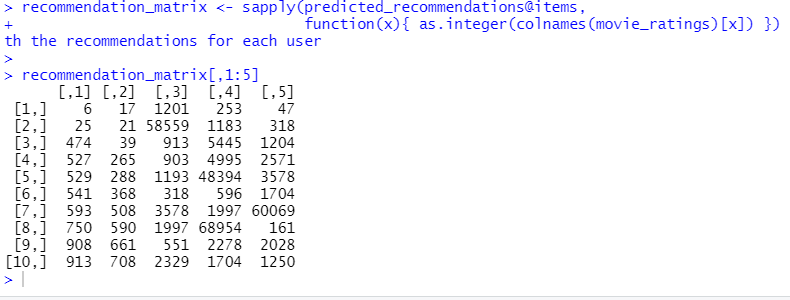
****

****

****

****

****

****